Fifth Excercise

The following program is about post-order iterator of a binary tree. In this program, a statck is used to save some temporay information. In the code of our presentation in class, each node will be pushed into and poped from the stack three times. Now ,please design a new algorithm to impement the post-order iterator where each node will be pushed into and poped from the stack only two times.

Most of the code is available, please fufil the function operator ++().

#include <iostream>

#include <QStack>

template <class Type> struct BTNode{

BTNode \*left;

Type data;

BTNode \*right;

BTNode(Type x){

data=x;

left=right=NULL;

}

};

template <class Type> class PostOrder;

template <class Type> class BinaryTree{

private:

BTNode<Type> \*root;

friend class PostOrder<Type>;

public:

BinaryTree(BTNode<Type>\*t){root=t;}

};

//Base class for BT Iterator

template <class Type> class TreeIterator {

protected:

const BinaryTree <Type> & T; //BT

const BTNode<Type> \*current;

public:

TreeIterator ( const BinaryTree <Type> & BT )

: T (BT), current (NULL) { }

virtual ~TreeIterator ( ) { }

virtual void First ( ) = 0;

virtual void operator ++ ( ) = 0;

operator bool(){return current!=NULL;}

const Type operator()()const{

if(current)

return current->data;

return (Type)0;

}

};

template <class Type> struct StkNode {

//Stack node definition

const BTNode <Type> \*Node; //Node Address

int PopTime; //Counter

StkNode ( BTNode <Type> \*N = NULL ) : Node (N), PopTime (0) { }

};

template <class Type> class PostOrder : public TreeIterator <Type> {

bool renew;

public:

PostOrder ( const BinaryTree <Type> & BT ):TreeIterator(BT){renew=false;}

~PostOrder ( ) { }

void First ( );

//Seek to the first node in postorder traversal

void operator ++ ( );

//Seek to the successor

protected:

QStack < StkNode<Type> > st; //Active record stack

};

template <class Type>

void PostOrder<Type>::First(){

current = T.root;

st.clear();

renew=true;

operator++();

}

template <class Type>

void PostOrder<Type>::operator ++(){

if(!renew && current==T.root && st.isEmpty()){

current=NULL;

return;

}

renew = false;

const BTNode<Type> \*p = current;

StkNode<Type> w;

do{

......

// Here, you must add necessary statements to set pointer current to the right posotion

}

while( p || !st.isEmpty( ));

}

int main(){

BTNode<int> \*p = new BTNode<int>(6);

p->left = new BTNode<int>(4);

p->right = new BTNode<int>(10);

p->left->left = new BTNode<int>(2);

p->right->left = new BTNode<int>(8);

p->right->right = new BTNode<int>(12);

BinaryTree<int> T(p);

PostOrder<int> it(T);

for(it.First();it;++it){

std::cout << it() << std::endl;

}

}